C: Remarks:

The applicants have corrected the claims 1 to fix technical and typographic errors. In addition, similar ommissions have been corrected in the specification where paragraph 12 belonged in paragraph 14 of the Summary of the Invention, and a word was omitted in paragraph 26.
The Claims 11-18 and 20-28 and 30 have been cancelled at this time such that the claims now presented are all method claims under the rules following the Devices for Medicine case. The computer system and program product claims will be prosecuted in a continuation application.

Reconsideration after this amendment is requested, and the examiner is respectfully traversed with respect to the rejections under 35 USC 103. The rejection assumes a teaching to one of ordinary skill of the art based on assumptions which are not factual, and therefore the rejection is still considered without merit.

A declaration of an inventor, Brian Robert Prasky, together with the attachments to the declaration are appended setting forth reasons why the examiner's basic assumptions were wrong in the previous office action, as well as the art remarks coorelating the BTAC and BTB nomenclature is again referenced as part of the record for reconsideration.

Reconsideration and allowance is now requested.

Looking at the art that the Examiner cites as the best art, US 6125444, most relevant to a claim directed the rejection fails to appreciate the contribution claimed to improving the performance of a pipeline architecture which when optomized has one instruction entering the pipeline every cycle, and so a delay or clearing negatively impacts the performance of the computer system. Branch history table logic has been used for prediction, and improvement of this logic may prevent delay in a pipeline and improve performance. The Examiner would say one skilled in the art would recognize that the prior art teaches the best way is described in US Patent 6125444 (naming Check, one the the inventors here, as teaching that best art). But the examiner also apparently thinks that one of ordinary skill in the art would also recognize that US6125444 should be improved by the teachings of Professors Patterson and Hennessy published three years prior to the filing of the US6125444. Why the examiner would think that is unclear, as the Professor's cited art was art owned by the assignee of all interest and employer of the inventors of US6125444, but Professors Patterson and Hennessy stated an admiration for use of Branch-Target Buffers, something employed in the machine described by the applicants. They stated, page 271 that the system needs to know from what address to fetch by the end of instruction fetch in the pipeline. That means, they said, we must know whether an as yet undecoded instruction is a branch, and if so, what the next PC (Program Counter) should be. If the instruction is a branch and "we" know what the next PC should be, "we" can have a branch penalty of zero. A branch-prediction cache that stores the predicted address for the next instruction after a branch is called a branch-target buffer or branch target cache. The professors like a two bit perditor but that has problems. However, IBM's POWERPC 620 (manufacturered by the assignee this application IBM) solves the problems of using a twobit predictor by providing a target buffer and a prediction buffer. The applicant's here provide a branch-target buffer and a branch history table for prediction.

The Examiner must have noted that the primary reference he cited here is summarized by its abstract. That US 6125444 says that it discloses "*A millimode capable computer system that provides control to millicode to allow the BHT operations to continue except when those special situations occur that require control of instruction fetch oporations must be provided and the BHT can be turned off for some sections of code execution, but not disabled for all. A single free running BHT functions for both a normal mode and a millimode for the centeral processor which can execute in millimode with a branch history table directing instruction fetch for which both a global BHT disable and millicode disables exist. Hit detection logic receives input from* *the global BHT disable, as well* ***as from an initialized control register bit and a processor control register bit to select the correct set target information and generate a*** *"branch history table hit detected" control signal."*

One must also note that the hit detection logic of the primary reference receives input from the global BHT disable, *as well as from* *an initialized control register bit and a processor control register bit to select the correct set target inormation* and this causes the hit detection logic to generate a "branch history table hit detected" control signal."

This "branch history table hit detected" control signal is then fed to the instruction fetching logic as a BHT Hit Detected control signal. This is shown in Figure 1.

But now, in order to reject the current claim, the Examiner argues that given the US US 6125444 method, the professors (who had previously already taught the older art) teach that "*in order to further increase the performance of branches a branch target buffer is often used, so that the target address can be calculated in the fetch stage instead of the decode stage (somewhere in pages 272-275). As the brach target buffer relies on having a prediction in order to determine the correct addresss, as seen by Figure 4.23 on page 275 (showing the steps for handling an instruction with a branch target buffer), if no prediction was able to be generated, because of a BHT being disabled, then it would have been obvious to one or ordinary skill in the art to also not use the BTB as it would have the data it needs to be made use of, therefore attempts to use a BTB in this situation would not only not make sense, but would be almost guaranteed to generate erroneous output. In addition, as can be seen by Column 2, lines 28-31 (presumeably of US 6125444)* *sensitive system operations require cache control, so for the same reason Check disables the BHT, the BHT would need to not be written to as well. Given the advantage of a BTB as disclosed by (Professor) Patterson, and the need to implement it in the system disclosed by Check, one of ordinary skill in the art at the time the invention was made would have been motivated to include a BTB, and also to disable its use when the BHT was diabled, as the entire branch prediciont mechanism must be diabled". This quoted paragraph is the Examiner's rejection of Claim 1.*

Now Column 2, lines 28-31 of US 6125444 says " One reason that the millicode would need to control when instruction fetches can occur is times when the millicode is doing sensitive operations tht require control of the type of cache functions that can occur."

Also, the one of ordinary skill in the art would read the professors statement on page 274 respecting "Figure 4.23 shows the steps followed when using a branch-target buffer and where these steps occur in the pipeline. From this we see that there will be no branch delay if a branch-prediction entry is found in the buffer and is correct. Otherwise there will be a penalty of at least two clock cycles. In practice this penalty could be larger, since the branch-target buffer must be updated. We could assume that the instruction following a branch or at the branch target is not a branch, and do the update during that instruciotn time; however, this does complicate the control. Instead, we will take a two-clock-cycle penalty wien the branch is not correctly predicted or when we get a miss in the buffer. Dealing with the mispredictions and misses is a significant chanllenge, since we typically will have to halt instruction fetch while we rewrite the buffer entry. Thus we would like to make this process fast to minimize penalty."

Also, the footnote to Figure 4.23 of the professors book describes *"The steps involved in handling an instruction with a branch target buffer. If the PC (Program Counter) of an instruction is found in the buffer, then the instruction must be a branch that is predicted taken; thus, fetching immediately begins from the predicted PC in ID (Instruction Decode). If the entry is not found and it subsequently turns out to be a taken branch, it is entered in the buffer along with the target which is known at the end of IG (Instruction Decode). If the entry is found, but the instruction turns out not to be a taken branch, it is removed from the buffer. If the instruction is a branch, if found, and is correctly predicted, then execution proceeds with no delays. If the prediction is incorrect, we suffer a on-clock-cycle delay fetching the wrong instruction and restart the fetch one clock clcyle later, leading to a total mispredict penalty of two clock cycles. If the branch is not found in the buffer and the instructions turns out to be a brand, we have proceeds as if the instrucion were a branch and can turn this into an assume-not-taken strategy. The penalty will differ depending on whether the branch is actually taken or not."*

The Examiner nevertheless, in his rejection, of claim 1 says *"As stated in the previous remarks, there is not hindsight reconstruction required to reject these claims and the Examiner asserts that his rejection and the way the reference are combined are the only way that one of ordinary skill in the art could possibly come to. Check teaches why the branch history table should be disabled, for areas whre the user explicitly does not want branch prediction to be used. The Examiner can see absolutely no reason, in the situation where the user explicity does not want branch prediction to occur, that the user would disable the BHT, but continue to use a BTB, a component that only exists for one purpose, to assist in branch prediction. To* *combine the reference in the way the applicant is arguing is proper, a user would have to diable the BHT and not use it, in order to avoid prediciton, but still for some reason decides he wants prediction still be used and then access the branch target buffer. Additionally, because the BTB relies upon input from the branch prediction mechanisms, it could not possible be correctly used, and if one was even to attempt to use a BTB with the Branch History Table Disabled, the BTB would output erroneous or random data, causing the computer to execute random instruction." The Examiner has asserted that "no one of ordinary skill in the art would ever come to this conclusion and that the only thing that makes sense in the context of this invention is to diable all components associated with branch prediction with the user has explicitly indicted that branch prediction is to be disabled, and given that a BTB has no purpose outside of branch prediction, and that cannout work with other branch prediction mechanisms disabled, that it would be diabled also."* See the Examiner's response to Arguments, paragraph 32.

But why should the applicants file for protection of an idea that was older than their own earlier development and one that the professors recognized came from their company and was still being used (the IBM PowerPC 620)? The Examiner apparently saw a need to create a straw man to reject. However, indeed, the applicants in this application in the background of the invention acknowledged that the prior art showed use of a branch history table (BHT) and a branch target buffer (BTB) in respect of predicting branches, see page 4, paragraph [0013].

The Examiner apparently did not understand the facts stated in the application, Page 4, paragraph [0013} which says that in using a BHT and a BTB in respect to predicting branches which exit state routines on a CISC microprocessor encounters the problem that such predictions can potentially corrupt the state of a machine thereby resulting in loss of data integrity. Thus, a clear need exists to allow such predictions where the exiting of a CISC based routing can be guaranteed not to have altered the integrity of the processed data outcomes based on system state.

Thus the Examiner's statement that *"no one of ordinary skill in the art would ever come to this conclusion and that the only thing that makes sense in the context of this invention is to diable all components associated with branch prediction with the user has explicitly indicted that branch prediction is to be disabled, and given that a BTB has no purpose outside of branch prediction, and that cannout work with other branch prediction mechanisms disabled, that it would be diabled also."* See the Examiner's response to Arguments, paragraph 32.

The Examiner failed to note that in the preferred embodiment of the claimed invention of present application that predictions are allowed to continue without diabling other branch prediction mechanisms.

It was in using the teaching of US 6125444 that inventors of that patent came to appreciate the need to improve the millicode operations of a CISC processor where the hit detection logic receives input from the global BHT disable, <u>*as well as from*</u> *an initialized control register bit and a processor control register bit to select the correct set target inormation* and this causes the hit detection logic to generate a "branch history table hit detected" control signal which stops the instructions fetching operations, as described in US6125444.

In the claimed invention, the shortcomings of the prior art are overcome and additional advantages are provided through the provision of a mechanism that prohibits certain branches to be predicted in an asychronous time frame in respect to the decoding of the said instruction. In particular, this modification is a descriptor bit in the opcode of a branch that states if the branch is allowed to be predicted or not. <u>By allowing a bit within the instruction text to state if a particular instance of a branch is to be written into the BTB, two results are achieved: 1) branches which are performance critical and do not return from state altering routines can be added into the BTB for branch prediction. 2) Branches which exit a routing which altered the state of a machine can be blocked from being written into the BTB such that they are never predicted. This invention allows for higher processor performance via branch prediction while maintaining data integrity and preventing a measurable growth in silocon area or power.</u>

Instead of using the Examiner's constructed prior art, as claimed in claim 1, the described this invention discovered in the claim as the claimed *"method operating a computer having a pipelined processor, comprising setting a bit within an instruction text field of a branch, said bit preventing the branch addesss from being placed into a branch history table buffer and into a branch target buffer to thereby prevent the branch from being written into the branch history table buffer and branch target buffer and* ***preventing the branch from being*** *predicted and to make the branch only detectable at the time frame of decode.*

The prior art does not provide for " *setting a bit within an instruction text field of a branch*" .

The prior art did something different as was stated in the abstract of the primary reference.

The primary reference US 6125444 uses a control signal to shut down fetching when the hit detection logic receives input from the global BHT disable, *as well as from an initialized control register bit and a processor control register bit to select the correct set target inormation* and this causes the hit detection logic to generate a "branch history table hit detected" control signal. While the examiner notes that the US6125444 addresses control of instruction fetches when the millicode is doing sensititve operations, Col. 2, lines 28-30, it should be noted that the mechanisms for doing this is to control branch history tables by entries into millicode control registers. Col. 2, lines 31-49. In a first use at the time of entry into a millicode routine certain millicode control registered are initialized which results in the Branch history table being diabled. A value in the control registers disables the branch history table and causes no fetching of the targets of entries. (Only) When code has passed sensitive parts of the system operations the millicode may turn the control bit in the control registers off and re-enable branch history table operations. Alternatively, the branch history table can be active upon entry into a millicode routine and when the sensitive function nears, the millicode may write a bit into a processor disable control register to disable the branch history table, and later when the sensitive area is passed the millicode is used to re-enable the branch history table.

The patent US6125444 continues Col. 2. Lines 50-col. 3 line 35 for a second use of the millicode control to disable brach history table function. When a very specific sequence of branching loops will be executed which hinders performance by sending the processor to the wrong spot or miss the desired branches. Then in the second use of millicode the millicode again writes a bit into the processor control register to diable the branch history table and to later after the specifal sequences have passes it re-enables the branch history table. In this use the routine must re-enable the function for otherwise the next millicode routine will also have the function disabled until millicode explicitly re-enables the funcationality of the branch history table.

This best teaching of the prior art did not perform a process of *setting a bit within an instruction text field of a branch* for *preventing the branch addesss from being placed into a branch history table buffer and into a branch target buffer to thereby prevent the branch from being written into the branch history table buffer and branch target buffer and* ***preventing the branch from being*** *predicted and to make the branch only detectable at the time frame of decode.*

This method of preventing a branch address brom being placed in the branch history table buffer addresses the problem that was heretofore encountered (by using the Primary Referenced US6125444) as explained in paragraph 13 of the description. It allows predictions where the exiting of a CISC based rounted can be guaranteed to not have altered the integrity of the process data outcomes based on system state.

Now the examininer argues that just provideing a branch target buffer in the branch history table buffer logic (a very fact acknowledged in the background of this application itself as to have been used in the prior art) would solve the problem being addressed because it meets the claimed subject matter.

But that is wrong. As the examiner says the prior art in the two uses of the control register would have not only turned the fetching mechanism off, that action would have disabled the branch history table and the branch target buffer. Page 12, Paragraph 32 says "Examiner can see absolutely no reason, in the situation where the user explicitly does not want branch prediction to occur, that the user would diable the BHT, but continue to use a BTB, a component that only exists for one purpose, to assist in branch prediction. Etc. (so) the only thing that makes sense in the context of this invention is to diable all components associated with branch prediciton when the user has explicitly indicated that branch perdiction is to be disabled, and given that a BTB has no purpose outside of branch prediction, and it cannot work with other brach prediciotn mechanisms disabled, that it would be diabled also." A reconstruction different from that actually preferred by the applicants.

However, as stated in the detailed description, the claimed subject matter does not operate like the prior art. It does not stop fetching or decoding. The present invention is derected to a method for selectively starting a decode where branches are classified as those branches which are predictable by the BHT/BTB and those which are not allowed to be predicted by the BHT/BTB. The claimed method (paragraph 26) allows taking a set of branches which were previously not allowed to be predicted by the BTB/BHB. As stated in Paragraph 28, the branch prediction logic is off searching for the next branch that it predicts the decode stage will encounter. It seqentially seraches the BTB for a branch address. Along with the branch address is the target address for the given branch. Additonally the Branch History Table stores states bits which predict if the branch should be guessdes taken or not taken. All these operations occur. When a taken branch is located, a fetch request is initated for the target and the invormation is passed to the decode. It is the decode which can block a target fetch which a BTB had caused to be kicked off at an early time fram. Thus because the fetch was kicked off earlier, the target can decode the cycle after the branch without incurring pipeline delay.

Thus clearly, *setting a bit within an instruction text field of a branch* for *preventing the branch addesss from being placed into a branch history table buffer and into a branch target buffer to thereby prevent the branch from being written into the branch history table buffer and branch target buffer and* ***preventing the branch from being*** *predicted and to make the branch only detectable at the time frame of decode*, was new and it has been discovered that this was a solution to difficulties found in the very art which the examiner references as the best prior art. By allowing a bit within the instruction text to state if a particular instance of a branch is to be written into the BTB, two results are achieved: 1) branches which are performance critical and do not return from state altering routines can be added into the BTB for branch prediction. 2) Branches which exit a routing which altered the state of a machine can be blocked from being written into the BTB such that they are never predicted. This invention allows for higher processor performance via branch prediction while maintaining data integrity and preventing a measurable growth in silocon area or power.

Now the changes made to claim 1 have been made to correct the undersigned's errorts, and the claim should be allowed.

The dependent claims are claims in combination with the specific method which is not shown in the art. They are stated because they are independently patentable in combination with the steps of claim 1. These claims are set out below to show how the art does not accomplish the claimed teachings.

2. (Original) A method as defined in claim 1 comprising predicting the direction and target of a branch prior to decode. Claim 2 states a process which occurs until encounter with an instruction text which acts as ***preventing the branch from being*** *predicted.*

3. (Original) A method as defined in claim 2 comprising predicting the direction and target of a branch prior to decode through a branch prediction array. Claim 3 states a process which allows fetching to occur prior to decode until encounter with an instruction text which acts after decode as ***preventing the branch from being*** *predicted.*

4. (Original) A method as defined in claim 1 comprising tracking the branch from the beginning of the pipe, decode, until the time frame that the given instruction is to be written into a branch prediction array. Claim 4 states a process which tracks the branch of the pipe and allows fetching and decoding to occur prior to decode of an instruction text which acts after decode as ***preventing the branch from being*** *predicted.*

5. (Original) A method as defined in claim 1 comprising denoting the instruction text field as a non-writable branch into the BTB. Claim 5 states a process which has an instruction text field as a non-writable branch into the BTB and upon decode of the instruction text it acts as ***preventing the branch from being*** *predicted.*

6. (Previously Amended) A method as defined in claim 5 comprising denoting the instruction field in the system area as a non-writable branch into the BTB in system whereby the branch is blocked from being written to the BTB. Claim 5 states a process which has an instruction text field as a non-writable branch into the BTB and upon decode of the instruction text it acts as ***preventing the branch from being*** *predicted and blocked from being written into the BTB.*

7. (Previously Amended) A method as defined in claim 5 comprising denoting the instruction field in the non-system area, the branch may be predicted via aliasing. Claim 5 states a process which has an instruction text field as a non-writable branch into the BTB and upon decode of the instruction text it acts as ***preventing the branch from being*** *predicted* and by this claimed feature of denoting the instruction field in the non-system area, the branch may be predicted via aliasing.

8. (Original) A method as defined in claim 1 wherein machine state altering code lies within an address range spanned by branch tag bits of the branch target buffer. The machine state altering

code lies within an address range spanned by branch tag bits of the branch target buffer function in combination with the *setting a bit within an instruction text field of a branch* for *preventing the branch addesss from being placed into a branch history table buffer and into a branch target buffer.*

10. (Original) The method as defined in claim 8 comprising denoting state altering code in the system area by a state bit within the BTB/BHT such that aliasing of branches is prevented within the system area. This claims 10 takes the step of denoting state altering code in the system area by a state bit within the BTB/BHT such that aliasing of branches is prevented within the system area where the branch could have been predicted via alising.

None of these features of claims 2 through 10 are present in the IBM PowerPC 620. The were not present in the IBM 390 system described by Check et al in the primary reference. Therefore it is not imaginable that anyone of ordinary skill in the art would create these out of thin air without looking at the detailed description described and claimed here.

It is respectfully submitted that the application should be in final condition for allowance which is respectfully requested.

RESPECTFULLY SUBMITTED
(For the inventors)
/Lynn L. Augspurger/
BY: Lynn L. Augspurger
Registration No. 24,227
Phone: 845-433-1174
Fax: 845-432-9601